Theoretical Computer Science 90 (1991) 81-93
Elsevier

An algebra of mixed computation

V.E. Itkin

Institute of Informatics Systems, Siberian Division of the USSR Academy of Sciences, Novosibirsk 630090, USSR

Abstract

Itkin, V.E., An algebra of mixed computation, Theoretical Computer Science 90 (1981), 81-93.

Algebraic tools for mixed computation are presented. Some axioms for informational objects, program functions, inputs and outputs are introduced. These axioms are sufficient for the correct mixed computation of some basic program composition forms.

1. Introduction

1.1. The historical information

The concept of mixed computation [1] was conceived for partial execution of programs whose result consists of two components: an intermediate memory state and a residual program.

Some introductory remarks from [2] are given below.

During a program execution, a situation occurs which allows some program fragments to be carried out of brackets (suspended) and separately executed.

For example, if the program is a sequence of statements $A_1; \cdots; A_5$ and there is no data flow between statements A_1, A_3 and A_2, A_4, then the statements A_1, A_3 can be suspended, which means that the sequence $A_2; A_4; A_5$ will be executed at first and then $A_1; A_3$.

In contrast with conventional or *normal* computation, the two-stage computation discussed above will be called *partitioned.* The first stage of a partitioned computation consists of *partial* computation and formation of a *residual* program. The second stage is the execution of the residual program. The first stage as a whole will be called a *mixed* computation of an initial program.

In general, a partitioned computation contains additional actions which provide the storage of information about suspended fragments. Moreover, a residual program may be modified before its execution. In particular, it may in turn be subjected to partitioned computation or optimization.

On the whole, mixed computation of a program can be treated as a conventional computation some parts of which are represented by not yet executed (suspended) program fragments.

The suspension of a fragment may be caused by an arbitrary decision to suspend certain fragments or may be forced by informational dependence of statements and logical conditions on previously suspended fragments (the latter is called a *forcible* suspension in contrast to an arbitrary suspension). Fragments situated on alternative branches of suspended logical conditions are also forcibly suspended.

A typical reason for an arbitrary suspension is the indeterminancy of value of some variable in a mixed computation. A suspension caused by lack of resources or slowness of the corresponding computations can also be treated as a suspension of this kind.

Mixed computation exploits the fact that some components of the informational structure of a program are separable. Partitioned computation allows separation of informational components that are intertwined by the control structure of a program. Thus, the transformation of a program in order to make informationally independent program components logically independent can be regarded as a static factorization of the program.

In [2], three algorithms *mix*, mix^0 and $\overline{mix}$ of mixed computation were developed. The *mix* algorithm is very natural but in general it is not correct. In [4] the correctness of *mix* was achieved by algebraic means. The algebra [4] is a realization of the transformational approach [3] to mixed computation.

The present paper is a modification of [4].

1.2. DMS-algebra

The algebra from [4] is a deterministic memory state one. Let R be a set of memory locations and C be a set of constants; then a memory state is a partial function s from R to C. The DMS-algebra constructed on (R, C) is defined as a frame

$$(2^R, \cap, \cup, \setminus; S, \mathrm{O}, +, *, \triangle)$$

where S is the set of all memory states; $\mathrm{O} \in S$, $\mathrm{O}(r)$ is undefined for any $r \in R$; $+ : S \times S \to S$; $*, \triangle : S \times 2^R \to S$; for any $s, s_1, s_2 \in S$, $Y \subseteq R$;

$d(s) = \{r \in R \mid s(r) \text{ is defined}\}$,
$d(s_1 + s_2) = d(s_1) \cup d(s_2)$,
if $r \in d(s_1)$ then $(s_1 + s_2)(r) = s_1(r)$,
if $r \notin d(s_1)$ and $r \in d(s_2)$ then $(s_1 + s_2)(r) = s_2(r)$,
$d(s * Y) = d(s) \cup Y$,
if $r \in d(s)$ and $r \in Y$ then $(s * Y)(r) = s(r)$,
$d(s \triangle Y) = d(s) \setminus Y$,
if $r \in d(s)$ and $r \notin Y$ then $(s \triangle Y)(r) = s(r)$.

Example. Let $R=\{r_1, r_2, r_3\}$, $C=\{1, 2, \ldots\}$; a memory state s is represented by a triple (a_1, a_2, a_3) where $a_i \in C \cup \{o\}$. For example, if $s=(3, 5, o)$ then $s(r_1)=3$, $s(r_2)=5$, $s(r_3)$ is undefined. Then we have

$$O=(o, o, o), \qquad (3, 5, o)+(o, 6, 8)=(3, 5, 8),$$

$$(3, 5, o) * \{r_2, r_3\}=(o, 5, o), \qquad (3, 5, o) \triangle \{r_2, r_3\}=(3, o, o).$$

Note. If R is an arbitrary set and $C=\{t, f\}$ then a memory state may be considered as a three-value predicate from R to $\{t, f, o\}$.

1.3. Operators

A notion of a program is modelled by an abstract "operator". An operator is defined as a frame

$$(p, \bar{p}, \text{in}(p), \text{out}(p), \text{yes}(p)),$$

where p is the name, $\bar{p}$ is the function (the partial function from S to S), $\text{in}(p) \subseteq R$ is the input, $\text{out}(p) \subseteq R$ is the complete output, $\text{yes}(p) \subseteq R$ is the obligatory output of the operator. Here we will call p an operator and write it down as

$$p=(\bar{p}, \text{in}(p), \text{out}(p), \text{yes}(p)).$$

If prog is an Algol-like program then yes(prog) is the set of locations which are assigned on any path of the program from begin to end.

An operator p is said to be *fundamental* iff it satisfies the following conditions (axioms):

$$\forall s \forall s' \quad (\bar{p}(s) \text{ is defined} \Rightarrow \bar{p}(s+s')=\bar{p}(s)+s'),$$

$$\forall s \quad (\bar{p}(s) \text{ is defined} \Rightarrow \bar{p}(s * \text{in}(p)) \text{ is defined}),$$

$$\forall s \quad (\bar{p}(s) \text{ is defined} \Rightarrow \bar{p}(s)=\bar{p}(s) * \text{out}(p)+s),$$

$$\forall s \quad (\bar{p}(s) \text{ is defined} \Rightarrow \text{yes}(p) \subseteq d(\bar{p}(s))).$$

A fundamental operator may be regarded as a closed information module.

Commentary to axioms. If $\bar{p}(s)$ is defined then

(0) an additional input information is passive; it is added to $\bar{p}(s)$ and neutralized by s and $\bar{p}(s)$:

$$\bar{p}(s)=\bar{p}(s+s)=\bar{p}(s)+s,$$

$$\bar{p}(s+s')=\bar{p}(s)+s'=(\bar{p}(s)+s)+s'=\bar{p}(s)+(s+s');$$

(1) $\bar{p}(s * \text{in}(p))$ is defined;

(2) $\bar{p}(s)$ is equivalent to s up to $\text{out}(p)$;

(3) $\bar{p}(s)(r)$ is defined for any $r \subset \text{yes}(p)$.

1.4. Explicators

For any $s \in S$ the $[s]$ operator is defined as

$$\overline{[s]}(s_1) = s + s_1 \quad \text{for any } s_1 \in S,$$

$$\text{in}([s]) = \emptyset,$$

$$\text{out}([s]) = \text{yes}([s]) = d(s).$$

Let us call $[s]$ the *explicator* of s. For example, if $s = \{r_1 = 4, r_2 = 8\}$ then $[s] = \{r_1 := 4, r_2 := 8\}$. It is true that the operator $[s]$ is fundamental for any $s \in S$.

1.5. Superposition of operators

Let $p_1 \cdot p_2$ be the name of an operator such that

$$\forall s \quad (\overline{p_1 \cdot p_2}(s) = \bar{p}_2(\bar{p}_1(s)))$$

$$\text{in}(p_1 \cdot p_2) = \text{in}(p_1) \cup (\text{in}(p_2) \backslash \text{yes}(p_1)),$$

$$\text{out}(p_1 \cdot p_2) = \text{out}(p_1) \cup \text{out}(p_2),$$

$$\text{yes}(p_1 \cdot p_2) = \text{yes}(p_1) \cup \text{yes}(p_2).$$

The operation "$\cdot$" is associative. It is true that if p_1 and p_2 are fundamental then $p_1 \cdot p_2$ is also fundamental.

1.6. Mixed computation

Mixed computation is defined [4] as a multi-valued mapping M from $P \times S$ to $P \times S$, where P is the set of all operators. If (p', s') is some value from $M(p, s)$, then p' is the residual program and s' is the intermediate memory state.

The transformation

$$(p, s) \to (p', s')$$

may be represented as

$$[s] \cdot p \to [s'] \cdot p'.$$

If $p' = p'' \cdot [s'']$, then s'' is the partial result of $M(p, s)$ and for any $s_0 \in S$,

$$\bar{p}(s + s_0) = s'' + \bar{p}''(s' + s_0).$$

An abstract example of the computation process of $M(p_1 \cdot p_2 \cdot p_3, s)$ is

$$\begin{aligned}
&[s] \cdot p_1 \cdot p_2 \cdot p_3 \\
&\quad \to [s_1] \cdot p_1 \cdot [s_2] \cdot p_2 \cdot p_3 \\
&\quad \to [s_1] \cdot p_1 \cdot [p_2(s_2)] \cdot p_3 \\
&\quad \to [s_1] \cdot p_1 \cdot [s_3] \cdot p_3 \cdot [s_4].
\end{aligned}$$

Here s_4 is the partial result and s_1 is the intermediate memory state.

In this process two sorts of transformations are used:

$$\frac{\bar{p}(s) \text{ is defined}}{[s]\cdot p \to [\bar{p}(s)]} \quad (\text{TRANS}_1)$$

$$[s]\cdot p \to [s']\cdot p\cdot [s''] \quad (\text{TRANS}_2)$$

where these expressions are considered on a metasyntactic level. The transformation TRANS_2 is important for mixed computation. If p is fundamental and

$$s' = s * (\text{in}(p) \cup (\text{out}(p) \backslash \text{yes}(p))),$$

$$s'' = s \triangle \text{out}(p),$$

then

$$\overline{[s]\cdot p} = \overline{[s']\cdot p\cdot [s'']}.$$

There are some possibilities for merging the operators and for parallel execution of transformations.

2. Axiomatization of DMS algebra

2.1. Algebra of nondeterministic memory states

A basis (of the memory) is a triple (R, C, T), where R is a set of memory locations, C is a set of constants and $T \subseteq R \times C$ is a set of permissible pairs.

An NDMS-algebra constructed on the basis (R, C, T) is a frame

$$(S, \text{O}, +, *, \triangle),$$

where $S = 2^T$ is the set of memory states (T is the total memory state); $\text{O} \in S$ is the empty subset of T (empty memory state); $+, *, \triangle : S \times S \to S$; for any $s, s_1, s_2 \in S$, $r \in R$,

$$s(r) = \{c \in C \mid (r, c) \in s\}, \qquad d(s) = \{r \in R \mid s(r) \neq \emptyset\},$$

$$D(s) = (d(s) \times C) \cap T, \qquad s_1 + s_2 = s_1 \cup (s_2 \backslash D(s_1)),$$

$$s_1 * s_2 = s_1 \cap D(s_2), \qquad s_1 \triangle s_2 = s_1 \backslash D(s_2).$$

For any $s_1, s_2 \in S$, $r \in R$,

$$(s_1 + s_2)(r) = \text{if } r \in d(s_1) \text{ then } s_1(r) \text{ else } s_2(r), \tag{1}$$

$$(s_1 * s_2)(r) = \text{if } r \in d(s_2) \text{ then } s_1(r) \text{ else O}, \tag{2}$$

$$(s_1 \triangle s_2)(r) = \text{if } r \in d(s_2) \text{ then O else } s_1(r), \tag{3}$$

Let $\text{gen}(Y) = (Y \times C) \cap T$.

$$\text{Then } D(s) = \text{gen}(d(s)). \tag{4}$$

For any $s, s_1, s_2 \in S$,

$$D(D(s)) = D(s), \tag{5}$$

$$D(s_1 \cup s_2) = D(s_1) \cup D(s_2) = D(s_1 + s_2) = D(s_1) + D(s_2), \tag{6}$$

$$D(s_1 \cap D(s_2)) = D(s_1) \cap D(s_2) = D(s_1 * s_2) = D(s_1) * D(s_2) \tag{7}$$

$$D(s_1 \setminus D(s_2)) = D(s_1) \setminus D(s_2) = D(s_1 \triangle s_2) = D(s_1) \triangle D(s_2), \tag{8}$$

For any $s_1, s_2 \in S$,

$$\exists s(s_2 = s_1 + s) \Leftrightarrow s_2 * s_1 = s_1 \Leftrightarrow s_1 + s_2 = s_2, \tag{9}$$

$$D(s_1) \subseteq D(s_2) \Leftrightarrow s_1 \subseteq D(s_2) \Leftrightarrow s_1 * s_2 = s_1. \tag{10}$$

For any $s \in S$,

$$D(s) = T * s. \tag{11}$$

Let $\mathrm{NOT}(s) = T \setminus D(s)$. Then for any $s_1, s_2 \in S$,

$$s_1 \triangle s_2 = s_1 * \mathrm{NOT}(s_2). \tag{12}$$

2.2. Quasi-Boolean algebra

A frame

$$(X, \mathrm{O}, +, *, \triangle)$$

(now the symbols O, +, *, △ have new meanings) is a quasi-Boolean algebra iff

(i) X is a nonempty set (of information elements);

(ii) $\mathrm{O} \in X$;

(iii) $+, *, \triangle : X \times X \to X$; these operations satisfy the following conditions (axioms): for any $x, y, z \in X$:

(A1) $(x + y) + z = x + (y + z)$,
(A2) $(x * y) * z = x * (y * z)$,
(A3) $x * (y * z) = x * (z * y)$,
(A4) $(x + y) * z = x * z + y * z$,
(A5) $z * (x + y) = z * x + z * y$,
(A6) $x * \mathrm{O} = \mathrm{O}$,
(A7) $\mathrm{O} * x = \mathrm{O}$,
(A8) $x * x = x$,
(A9) $x + y * x = x$,
(A10) $x \triangle x = \mathrm{O}$,
(A11) $x * y + x \triangle y = x$,
(A12) $x * y + z \triangle y = z \triangle y + x * y$,
(A13) $(x + y) \triangle z = x \triangle z + y \triangle z$,
(A14) $x \triangle (y + z) = (x \triangle y) \triangle z$,
(A15) $(x * y) \triangle z = x * (y \triangle z)$,
(A16) $(x * y) \triangle z = (x \triangle z) * y$.

Theorem 1. *For any basis of memory, an NDMS-algebra constructed on this basis is quasi-Boolean.*

It is true that the (A1)-(A16) axiom system is the complete axiomatization of equality in any NDMS-algebra (for any basis of memory) [5].

Further through the text we suppose some quasi-Boolean algebra $(X, 0, +, *, \triangle)$ to be fixed.

Let, by definition, for any $x, y \in X$

$$x \leqslant y \Leftrightarrow \exists z(y = x + z).$$

For any $x, x_0, y, y_0, z, z_1, z_2 \in X$,

$$x + x = x, \tag{13}$$

$$x * y + x = x, \tag{14}$$

$$x \leqslant y \Leftrightarrow y * x = x \Leftrightarrow x + y = y, \tag{15}$$

$$x \leqslant x. \tag{16}$$

$$x \leqslant y \text{ and } y \leqslant z \Rightarrow x \leqslant z, \tag{17}$$

$$x \leqslant y \text{ and } y \leqslant x \Rightarrow x = y, \tag{18}$$

$$x + x * y = x, \tag{19}$$

$$x \leqslant y \Rightarrow y + x = y, \tag{20}$$

$$x * y = 0 \Rightarrow z_1 * x + z_2 * y = z_2 * y + z_1 * x, \tag{21}$$

$$x + x \triangle y = x \triangle y + x, \tag{22}$$

$$x * y \leqslant x, \tag{23}$$

$$x \leqslant y \Rightarrow x * z \leqslant y * z, \tag{24}$$

$$x \leqslant y \Rightarrow z + x \leqslant z + y, \tag{25}$$

$$y \leqslant z \Rightarrow x * x_0 + y * x_0 \leqslant x + z * x_0, \tag{26}$$

$$x * y = 0 \Rightarrow z * y \leqslant z \triangle x, \tag{27}$$

$$x * y + y * x = x * y, \tag{28}$$

$$x * y + z \triangle x = z \triangle x + x * y, \tag{29}$$

$$x \triangle y + y \triangle x = y \triangle x + x \triangle y, \tag{30}$$

$$x + y + x = x + y, \tag{31}$$

$$x + y = x + y \triangle x, \tag{32}$$

$$x * z + y * z + x = x + y * z, \tag{33}$$

$$x + y + x * z = x + y, \tag{34}$$

$$x+y+z*x=x+y, \tag{35}$$

$$x+(x+y)*z=(x+y)*z+x, \tag{36}$$

$$y\leqslant x\triangle z \Rightarrow y*z=0, \tag{37}$$

$$x\leqslant y \Rightarrow x+(y+y_0)*z=(y+y_0)*z+x, \tag{38}$$

$$x*(y+z)=x*(z+y), \tag{39}$$

$$(x\triangle y)*(x\triangle z)=(x\triangle z)*(x\triangle y), \tag{40}$$

$$x\triangle y+x\triangle z=x\triangle z+x\triangle y, \tag{41}$$

$$(x\triangle y)\triangle z=(x\triangle z)\triangle y, \tag{42}$$

$$x\triangle(y*z)=x\triangle(z*y), \tag{43}$$

$$x\triangle(y*z)=x\triangle y+x\triangle z, \tag{44}$$

$$(x\triangle y)\triangle(y\triangle z)=x\triangle y, \tag{45}$$

$$x\triangle(y\triangle x)=x, \tag{46}$$

$$(x*z)\triangle(y\triangle z)=x*z, \tag{47}$$

$$(y*z)*(y\triangle z)=0, \tag{48}$$

$$(y*z)\triangle(y\triangle z)p=y*z, \tag{49}$$

$$x\triangle(y\triangle z)=x*y*z+x\triangle y, \tag{50}$$

$$x\triangle(y\triangle z)=x*z+x\triangle y, \tag{51}$$

$$x*y=0 \Leftrightarrow y*x=0. \tag{52}$$

3. Fundamental operators and recognizers

3.1. Fundamental operators

Let for any partial functions f, f_1, f_2 from X to X, for any $x, y\in X$,

$$\#(f,x) \Leftrightarrow^{\mathrm{df}} f(x) \text{ is defined;}$$

$$f_1(x)==f_2(y) \Leftrightarrow^{\mathrm{df}} (\#(f_1,x) \Leftrightarrow \#(f_2,y)) \\ \& (\#(f_1,x) \Rightarrow f_1(x)=f_2(y)).$$

An operator is a frame

$$(f,x,y,z),$$

where f is a partial function from X to X; $x, y, z\in X$.

If the operator (f,x,y,z) is marked by p, then f, x, y, z are marked by $\bar{p}$, in(p), out(p), yes(p), respectively.

We state that for any operators p_1, p_2

$$p_1 = p_2 \Leftrightarrow^{\mathrm{df}} \bar{p}_1 = \bar{p}_2 \;\&\; \mathrm{in}(p_1) = \mathrm{in}(p_2) \;\&\; \mathrm{out}(p_1) = \mathrm{out}(p_2) \;\&\; \mathrm{yes}(p_1) = \mathrm{yes}(p_2).$$

An operator p is said to be *fundamental* iff the following properties hold:

(F0) $\forall x \forall y(\#(\bar{p}, x) \Rightarrow \bar{p}(x+y) = p(x) + y)$,

(F1) $\forall x(\#(\bar{p}, x) \Rightarrow \#(\bar{p}, x * \mathrm{in}(p)))$,

(F2) $\forall x(\#(\bar{p}, x) \Rightarrow \bar{p}(x) = \bar{p}(x) * \mathrm{out}(p) + x)$,

(F3) $\forall x(\#(\bar{p}, x) \Rightarrow \mathrm{yes}(p) * p(x) = \mathrm{yes}(p))$.

For any operator p,

$$\mathrm{F0}(p) \Rightarrow \forall x(\#(\bar{p}, x) \Rightarrow \bar{p}(x) = \bar{p}(x) + x), \tag{53}$$

$$\mathrm{F0}(p) \Rightarrow \forall x \forall y(\#(\bar{p}, x) \;\&\; x \leq y \Rightarrow \#(\bar{p}, y) \;\&\; \bar{p}(x) \leq \bar{p}(y)). \tag{54}$$

4. Composition of operators

Let $p_1 \cdot p_2$ be such an operator that

$$\forall x(\overline{p_1 \cdot p_2}(x) = = \bar{p}_2(\bar{p}_1(x))),$$

where $\#(\overline{p_1 \cdot p_2}, x)$ iff $\#(\bar{p}_1, x)$ and $\#(\bar{p}_2, \bar{p}_1(x))$,

$$\mathrm{in}(p_1 \cdot p_2) = \mathrm{in}(p_1) + \mathrm{in}(p_2 \triangle \mathrm{yes}(p_1)),$$

$$\mathrm{out}(p_1 \cdot p_2) = \mathrm{out}(p_2) + \mathrm{out}(p_1),$$

$$\mathrm{yes}(p_1 \cdot p_2) = \mathrm{yes}(p_2) + \mathrm{yes}(p_1).$$

The operation "$\cdot$" is associative.

Theorem 2. *For any fundamental operators p, q the operator $p \cdot q$ is fundamental.*

Proof. Proof of $\mathrm{F1}(p \cdot q)$. We have to prove $\#(\overline{p \cdot q}, x * \mathrm{in}(p \cdot q))$, i.e.

$$\#(\bar{p}, x * \mathrm{in}(p \cdot q)) \;\&\; \#(\bar{q}, \bar{p}(x * \mathrm{in}(p \cdot q))).$$

$\#(\bar{p}, x)$ and $\mathrm{F1}(p)$ imply $\#(\bar{p}, x * \mathrm{in}(p))$; furthermore, $\mathrm{in}(p) \leq \mathrm{in}(p \cdot q)$ and (54) imply $\#(\bar{p}, x * \mathrm{in}(p \cdot q))$.

Now we have to prove $\#(\bar{q}, \bar{p}(x * \mathrm{in}(p \cdot q)))$. $\#(\bar{q}, \bar{p}(x))$ and $\mathrm{F1}(q)$ imply $\#(\bar{q}, \bar{p}(x) * \mathrm{in}(q))$; (54) for q and

$$\bar{p}(x) * \mathrm{in}(q) \leqslant \bar{p}(x * \mathrm{in}(p \cdot q)) \tag{55}$$

imply $\#(\bar{q}, \bar{p}(x * \mathrm{in}(p \cdot q)))$.

Now we have to prove (55). $\#(\bar{p}, x)$ and $\mathrm{F1}(p)$ imply $\#(\bar{p}, x * \mathrm{in}(p))$; $\mathrm{F0}(p)$ implies $\bar{p}(x) = \bar{p}(x * \mathrm{in}(p)) + x \triangle \mathrm{in}(p)$.

We have

$$\bar{p}(x) * \mathrm{in}(q) = \bar{p}(x * \mathrm{in}(p)) * \mathrm{in}(q) + (x \triangle \mathrm{in}(p)) * \mathrm{in}(q). \tag{56}$$

The equality

$$\bar{p}(x * \mathrm{in}(p \cdot q)) = \bar{p}(x * \mathrm{in}(p)) + x * \mathrm{in}(q) \tag{57}$$

is true since

$$\begin{aligned}
&\bar{p}(x * \mathrm{in}(p \cdot q)) = \quad \{\mathrm{F0}(p)\} \\
&\quad = \bar{p}(x * \mathrm{in}(p)) + x * (\mathrm{in}(q) \triangle \mathrm{yes}(p)) = \quad \{\mathrm{F3}(p)\} \\
&\quad = \bar{p}(x * \mathrm{in}(p)) + (x * \mathrm{in}(q)) * \mathrm{yes}(p) + (x * \mathrm{in}(q)) \triangle \mathrm{yes}(p) \\
&\quad = \bar{p}(x * \mathrm{in}(p)) + x * \mathrm{in}(q).
\end{aligned}$$

Relation (55) is equivalent to having the relation $\leqslant$ between the right-hand sides of (56) and (57). This can be derived from relation (26) by some substitutions.

Proof of $\mathrm{F2}(p \cdot q)$. Let $\#(\overline{p \cdot q}, x)$. Then

$$\begin{aligned}
&\overline{p \cdot q}\,(x) = \bar{q}(\bar{p}(x)) = \quad \{\mathrm{F2}(q)\} \\
&\quad = \bar{q}(\bar{p}(x)) * \mathrm{out}(q) + \bar{p}(x) =^1 \\
&\quad = \bar{q}(\bar{p}(x)) * \mathrm{out}(q) + \bar{q}(\bar{p}(x)) * \mathrm{out}(p) + \bar{p}(x) =^2 \\
&\quad = \bar{q}(\bar{p}(x)) * \mathrm{out}(q) + \bar{q}(\bar{p}(x)) * \mathrm{out}(p) + x = \\
&\quad = \bar{q}(\bar{p}(x)) * (\mathrm{out}(q) + \mathrm{out}(p)) + x.
\end{aligned}$$

Let us denote $\bar{q}(\bar{p}(x))$ by y.

Proof of $=^1$.

$$\begin{aligned}
&y * \mathrm{out}(q) + y * \mathrm{out}(p) + \bar{p}(x) = \quad \{\mathrm{F2}(q)\} \\
&\quad = y * \mathrm{out}(q) + (y * \mathrm{out}(q) + \bar{p}(x)) * \mathrm{out}(p) + \bar{p}(x) \quad = \{\mathrm{F2}(p)\} \\
&\quad = y * \mathrm{out}(q) + \bar{p}(x).
\end{aligned}$$

Proof of $=^2$.

$$\begin{aligned}
&y * \mathrm{out}(p) + \bar{p}(x) = \quad \{\mathrm{F2}(p)\} \\
&\quad = y * \mathrm{out}(p) + \bar{p}(x) * \mathrm{out}(p) + x = \\
&\quad = (y + \bar{p}(x)) * \mathrm{out}(p) + x = \quad \{(53) \text{ for } q\} \\
&\quad = y * \mathrm{out}(p) + x. \qquad \square
\end{aligned}$$

5. Information connections between operators

If for any operators p and q

$$\mathrm{out}(p) * \mathrm{in}(q) \neq 0,$$

then we say that there is an *information connection* from p to q. The relations

$$\mathrm{out}(p) * \mathrm{in}(q) = 0, \qquad \mathrm{out}(p) * \mathrm{out}(q) = 0$$

are denoted by p NI q, p NO q respectively.

For any fundamental operator p, for any $x \in X$,

$$\#(\bar{p}, x) \Leftrightarrow \#(\bar{p}, x * \mathrm{in}(p)), \tag{58}$$

$$\bar{p}(x) = \bar{p}(x * \mathrm{in}(p)) + x, \tag{59}$$

$$\#(\bar{p}, x) \Rightarrow \bar{p}(x) \triangle \mathrm{out}(p) = x \triangle \mathrm{out}(p). \tag{60}$$

For any fundamental operators p and q, for any $x \in X$, if p NI q then

$$\#(\bar{p}, x) \Rightarrow x * \mathrm{in}(q) = \bar{p}(x) * \mathrm{in}(q), \tag{61}$$

$$\#(p, x) \Rightarrow \bar{p}(x) * \mathrm{in}(q) \leqslant x \triangle \mathrm{out}(p), \tag{62}$$

$$\overline{p \cdot q}\,(x) = = \bar{q}(x * \mathrm{in}(q)) + p(x), \tag{63}$$

$$\overline{p \cdot q}\,(x) = = \bar{q}(x \triangle \mathrm{out}(p)) + \bar{p}(x), \tag{64}$$

$$\overline{p \cdot q}\,(x) = = (\bar{q}(x * \mathrm{in}(q))) * \mathrm{out}(q) + (\bar{p}(x * \mathrm{in}(p))) * \mathrm{out}(p) + x, \tag{65}$$

$$\mathrm{out}(q) = \mathrm{yes}(q) \Rightarrow \overline{p \cdot q}\,(x) = = \bar{q}(x) * \mathrm{out}(p) + \bar{p}(x), \tag{66}$$

$$q \text{ NI } p \;\&\; p \text{ NO } q \Rightarrow \overline{p \cdot q}\,(x) = = \overline{q \cdot p}\,(x). \tag{67}$$

The relations (63)-(66) may be used for parallel computation of $\bar{p}_2(\bar{p}_1(x))$.

6. Mixed computation

6.1. Explicators

For any $x \in X$ we define the operator $[x]$ as follows:

$$\forall y(\overline{[x]}(y) = x + y), \qquad \mathrm{in}([x]) = 0, \quad \mathrm{out}([x]) = \mathrm{yes}([x]) = x.$$

The operator $[x]$ is called the *explicator* of x.

Note. It would be natural to state $\mathrm{out}([s]) = \mathrm{yes}([s]) = D(s)$ in NDMS-algebra. But s plays the role of $D(s)$ when s is used as the right-hand side argument of operation "$*$".

It is true that the operator $[x]$ is fundamental for any $x \in X$; for any $x, y \in X$:

$$[x] \text{ NI } [y], \qquad [x] \cdot [y] = [y + x].$$

6.2. Principal statement

Theorem 3. *For any fundamental operator p, for any* $x \in X$,

$$\overline{[x] \cdot p} = \overline{[x'] \cdot p \cdot [x'']}, \tag{68}$$

$$\mathrm{in}([x] \cdot p) = \mathrm{in}([x'] \cdot p \cdot [x'']), \tag{69}$$

$$\mathrm{out}([x] \cdot p) = \mathrm{out}([x'] \cdot p \cdot [x'']), \tag{70}$$

where

$$x' = x * (\mathrm{in}(p) + \mathrm{out}(p) \triangle \mathrm{yes}(p)),$$

$$x'' = x \triangle \mathrm{out}(p).$$

Proof. Proof of (68). We denote $\mathrm{in}(p) + \mathrm{out}(p) \triangle \mathrm{yes}(p)$ by z. For any $y \in X$,

$$\overline{[x'] \cdot p \cdot [x'']}(y) ==$$

$$== \overline{p \cdot [x'']}(x * z + y) == \quad \{(58), (59)\}$$

$$== \overline{p \cdot [x'']}((x * z + y) * \mathrm{in}(p)) + x * z + y =^{1} =$$

$$== \overline{p \cdot [x'']}((x + y) * \mathrm{in}(p)) + x * z + y ==$$

$$== x \triangle \mathrm{out}(p) + \bar{p}((x + y) * \mathrm{in}(p)) + x * z + y =^{2} =$$

$$== \bar{p}((x + y) * \mathrm{in}(p)) + x \triangle \mathrm{out}(p) + x * z + y == \quad \{\mathrm{F3}(p)\}$$

$$== \bar{p}((x + y) * \mathrm{in}(p)) + (x * \mathrm{out}(p)) * \mathrm{yes}(p) + x \triangle \mathrm{out}(p) + x * z + y =^{3} =$$

$$== \bar{p}((x + y) * \mathrm{in}(p)) + (x + y) == \quad \{(58), (59)\}$$

$$== \bar{p}(x + y) == p(\overline{[x]}(y)) == \overline{[x] \cdot p}(y).$$

Proof of $=^{1}=$.

$$(x * z + y) * \mathrm{in}(p) =$$

$$= (x * (\mathrm{in}(p) + \mathrm{out}(p) \triangle \mathrm{yes}(p)) + y) * \mathrm{in}(p) =$$

$$= (x + x * (\mathrm{out}(p) \triangle \mathrm{yes}(p))) * \mathrm{in}(p) + y * \mathrm{in}(p) =$$

$$= x * \mathrm{in}(p) + y * \mathrm{in}(p) = (x + y) * \mathrm{in}(p).$$

Proof of $=^{2}=$. Denote $(x + y) * \mathrm{in}(p)$ by w. Then

$$x \triangle \mathrm{out}(p) + \bar{p}(w) ==$$

$$== x \triangle \mathrm{out}(p) + \bar{p}(w) * \mathrm{out}(p) + \bar{p}(w) \triangle \mathrm{out}(p) == \quad \{(60)\}$$

$$== \bar{p}(w) * \mathrm{out}(p) + x \triangle \mathrm{out}(p) + w \triangle \mathrm{out}(p) ==$$

$$== \bar{p}(w) * \mathrm{out}(p) + (x + w) \triangle \mathrm{out}(p) == \quad \{(36)\}$$

$$== \bar{p}(w) * \mathrm{out}(p) + (w + x) \triangle \mathrm{out}(p) == \quad \{(60)\}$$

$$== \bar{p}(w) * \mathrm{out}(p) + \bar{p}(w) \triangle \mathrm{out}(p) + x \triangle \mathrm{out}(p) ==$$

$$== \bar{p}(w) + x \triangle \mathrm{out}(p).$$

Proof of $=^3=$.

$$(x * \text{out}(p)) * \text{yes}(p) + x \triangle \text{out}(p) + x * \text{in}(p) + x * (\text{out}(p) \triangle \text{yes}(p)) =$$

$$= x * \text{in}(p) + x \triangle \text{out}(p) + (x * \text{out}(p)) * \text{yes}(p) + (x * \text{out}(p)) \triangle \text{yes}(p) =$$

$$= x * \text{in}(p) + x \triangle \text{out}(p) + x * \text{out}(p) =$$

$$= x * \text{in}(p) + x = x.$$

We denote $[x] \cdot p, [x'] \cdot p \cdot [x'']$ by p_1, p_2, respectively.
Proof of (69).

$$\text{in}(p_2) = \text{in}(p) \triangle (x * (\text{in}(p) + \text{out}(p) \triangle \text{yes}(p))) =$$

$$= (\text{in}(p) \triangle (x * \text{in}(p))) \triangle (x * (\text{out}(p) \triangle \text{yes}(p))) =$$

$$= (\text{in}(p) \triangle x) \triangle (x * (\text{out}(p) \triangle \text{yes}(p))) =$$

$$= \text{in}(p) \triangle (x + x * (\text{out}(p) \triangle \text{yes}(p))) = \text{in}(p) \triangle x = \text{in}(p_1).$$

Proof of (70).

$$\text{out}(p_2) = x \triangle \text{out}(p) + \text{out}(p) + x * \text{in}(p) + x * (\text{out}(p) \triangle \text{yes}(p)) =$$

$$= \text{out}(p) + x + x * \text{in}(p) + x * (\text{out}(p) \triangle \text{yes}(p)) =$$

$$= \text{out}(p) + x = \text{out}(p_1).$$

7. Problems

Note some problems for further investigations:

- Specification of programs by mixed computation algebras.
- Expanding NDMS and quasi-Boolean algebras by introducing D, T and NOT.
- Introduction of arrays as informational elements.
- Representation of explicator systems.
- Design of transformational semantics of program languages which would allow us to use the reserves of partial and parallel computation.

References

[1] A.P. Ershov, On the partial computation principle, *Inform. Process. Lett.* **6**(2) (1977) 38-41.

[2] A.P. Ershov and V.E. Itkin, Correctness of mixed computation in Algol-like programs, Lecture Notes in Computer Science **53** (Springer, Berlin, 1977) 59-77.

[3] A.P. Ershov, Mixed computation: potential applications and problems for study, *Theoret. Comput. Sci.* **18**(1) (1982) 41-67.

[4] V.E. Itkin, An algebra and axiomatization system of mixed computation, in: D. Bjørner, A.P. Ershov and N.D. Jones, eds., *Partial Evaluation and Mixed Computation* (North-Holland, Amsterdam, 1988) 209-224.

[5] V.E. Itkin, An axiomatization of the equality in an algebra of nondeterministic memory states, in: *Practical and Theoretical Programming Methods* (Novosibirsk, 1987) 5-19 (in Russian).